平成１６年度

九州地区国立大学法人等技術専門職員研修分科会資料

Ｃ言語による機器の制御と光ＣＴの基礎実験

会場　：　大分大学工学部　電気電子工学科電子コース

2004年9月9日

1　概要

9 月 9 日はC言語による簡単なシミュレーションと機器の制御をテーマとしています。前半は，C 言語を用いたプログラミングによって、運動方程式の数値的な解法を試みます。これまで C 言語でプログラムを作成した経験のない方が対象です。後半ではコンピューターの果たしている他の役割として ”制御 ”を取りあげます。USB 機器を利用してラジコンカーを走らせます。

注

- 全角文字と半角文字の使い分けに注意してください。全角文字は使用できる場所が限られていて，メッセージの表示等でしか使用できません。
- 全角のスペースとタブキーを用いてインデント（字下げ）をするとエラーになります。

2　C++ BuilderX を用いる

2.1　はじめに

ここで使用する言語は Borland 社 (http://www.borland.co.jp) の C++BuilderX Personal を用います。これは C/C++言語を用いたプログラミング環境を提供するソフトウエアで、統合開発環境 (IDE) と呼ばれるものです。はじめに、デスクトップ上にある C++BuilderX のアイコンをダブルクリックして起動します（図 1)。

図 1: C++BuilderX のアイコン

初めての起動であっても、何らかのファイルが読み込まれて起動します。２回目以降は前回作成したプログラムが読み込まれて起動します。メインメニューの［ファイル］［プロジェクトを閉じる］で閉じてしまいます（図 2)。

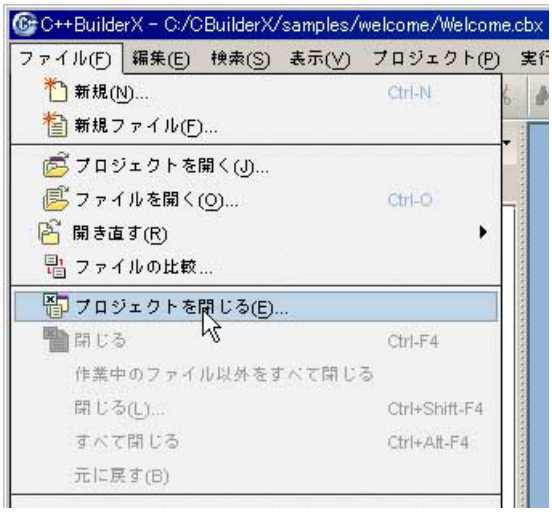


図 2: プロジェクトを閉じる

図 3: 閉じるプロジェクトを選択する

すると図 3 のような確認画面が現れますので、チェッックして全部消してしまいます。こうして、図 4 のような何のファイルも開かれていない画面になります。

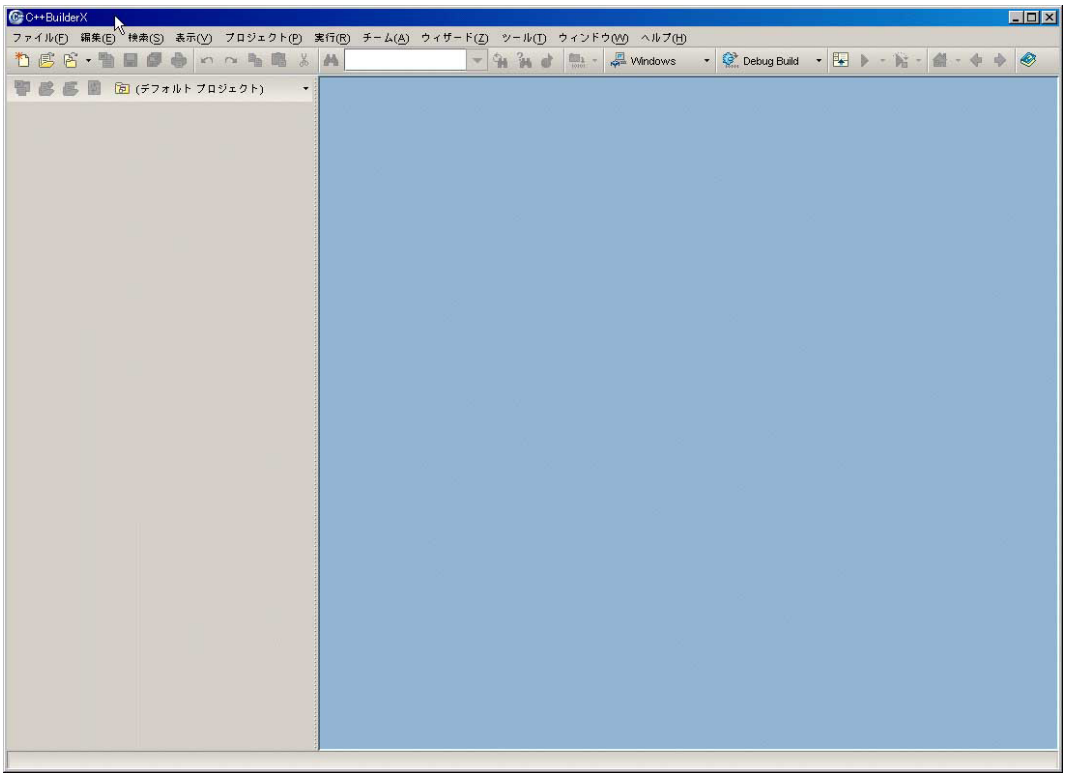


図 4: 何も開かれていない画面

2.2 プロジェクトを開始する

メインメニューから［ファイル］［新規］を選択すると図 5 の窓が表示されます。

図 5: 新規コンソール

図 6: プロジェクトの作成

図 5 は作成するプログラムの種類を選ぶ画面で、ここでは新規コンソールを選びます。OK をクリックすると図 6 が表示されます。この図にある”プロジェクト”とはプログラム作成の作業全体を意味します。図 6 でプロジェクトに適当な名前を付けると、ホームディレクトリーの中にフォルダーが作成されます。ここでは Administrator というユーザーで、”first”というプロジェクトですから、下記のフォルダーが作成されます (それぞれのユーザー名に置き換えること)。

```
C:\Documents and Settings\Administrator\cbproject\first
```

図 6 で［次へ］をクリックし、図 7 のプラットフォームの選択画面を表示します。Windows 用のソフトを作成するので初期設定のまま次へをクリックすると、図 8 の画面になり、untitled にチェックを入れて終了をクリックすると、first というプロジェクトの初期設定が完了します (図 9)。

図 7: プラットフォームの選択

図 8: default のソースを選ぶ

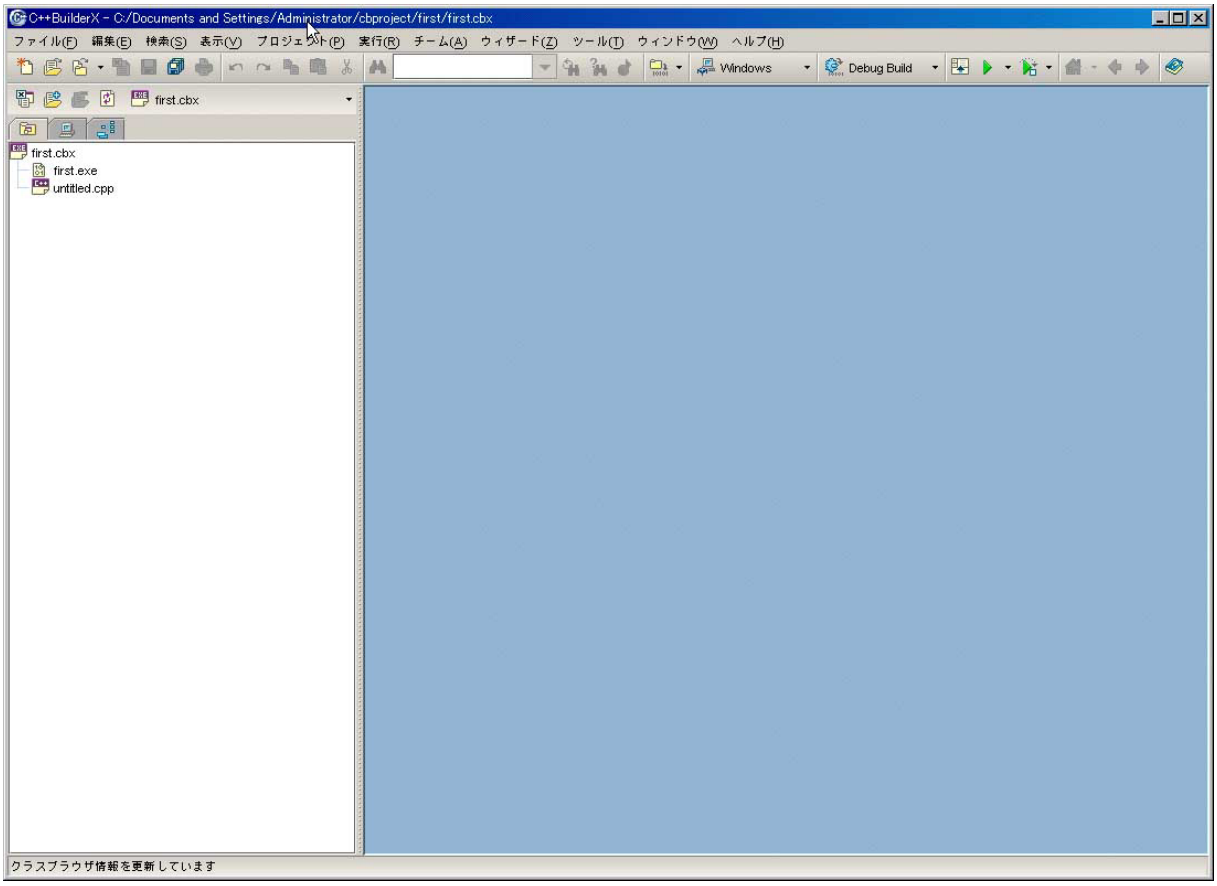

C++BuilderX - C:/Documents and Settings/Administrator/cbproject/first/first.cbx
ファイル(F) 編集(E) 検索(S) 表示(V) プロジェクト(P) 実行(R) チーム(A) ウィザード(Z) ツール(T) ウィンドウ(W) ヘルプ(H)
Windows
Debug Build
first.cbx
first.cbx
first.exe
untitled.cpp
クラスブラウザ情報を更新しています

図 9: 初期設定の終了

2.3 プログラムの作成

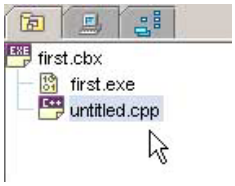
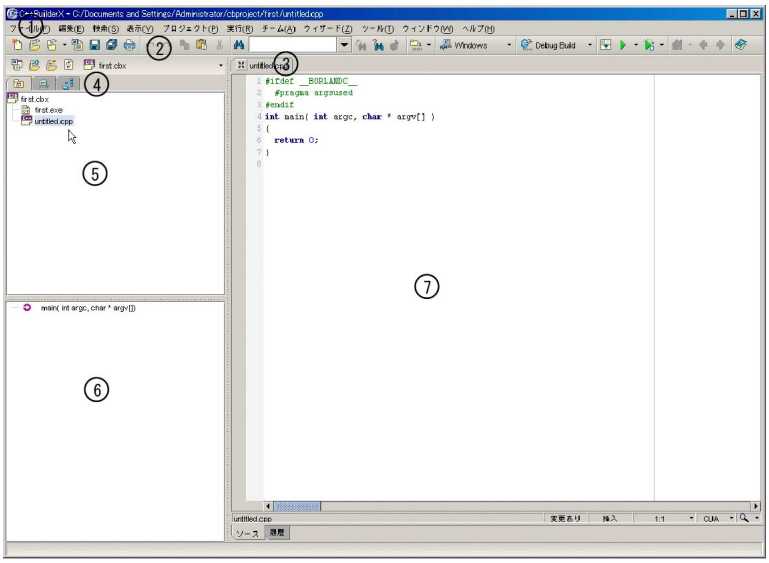


図 10: untitled.cpp

図 10 に示す untitled.cpp という文字をダブルクリックすると図 11 の画面が表示されます。以下でこの図にあらわれている領域の（四角い枠の）名称をあげます。



① メインメニュー
② ツールバー
③ ファイルタブ
④ プロジェクトツールバー
⑤ プロジェクトペイン
⑥ 構造ペイン
⑦ 内容ペイン

図 11: 領域の名称

最初に簡単な C 言語のプログラムを作成するため、一端⑦の内容ペインのテキストをすべて削除します。その後、図 12 のように記述して、ツールバーから［すべて保存］をクリックします（以下、必要に応じて保存すること）。

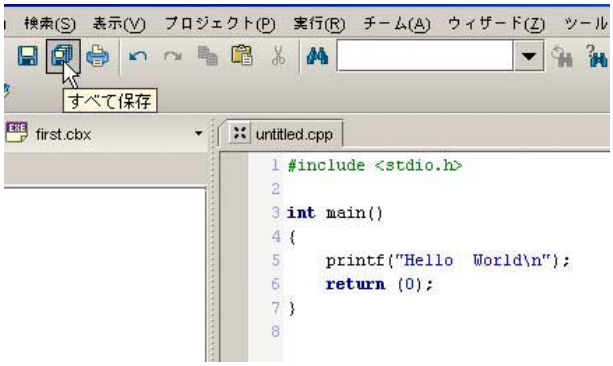


図 12: Hello World

このプログラムは画面上に”Hello World”と表示して 1 行改行するプログラムです。以下、簡単に説明すると、5 行目の関数 `printf` が画面に描画する働きをします。5 行目にある`\n` は改行を表します。この `printf` を使用するために 1 行目で、`#include <stdio.h>` と記述します。`stdio.h` は標準的な入出力を定義したパッケージで、その中で `printf` も定義されています。5、6 行目に示すように、C ではセミコロン (`;`) を用いて文の終わりを示します。

2.4　プログラムのコンパイルと実行

先に入力したプログラムをコンパイラを使用して実行可能プログラムにします（コンパイルするという）。

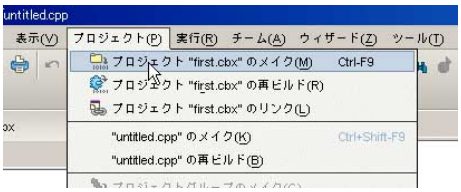


図 13: プログラムのコンパイル

メインメニューの［プロジェクト］［プロジェクト”first.cbx”のメイク］を実行すると（図 13）、window 下部にメッセージペインという窓が開きます（図 14）。ここにはエラーなどのメッセージが表示されます。エラーメッセージが有れば、プログラムを修正します。

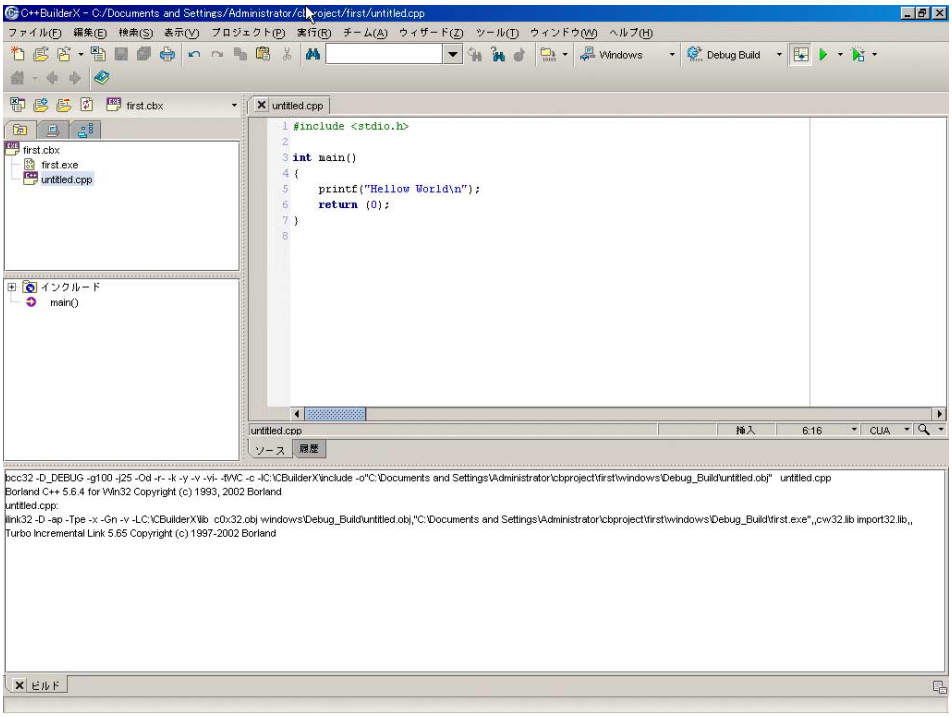


図 14: メッセージペイン

コンパイルが終了したら、ツールバーの［プロジェクトの実行］をクリックします (図 15)。

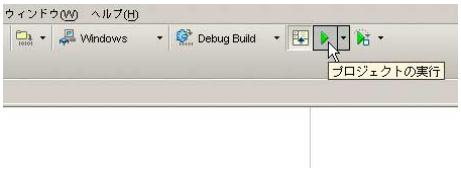


図 15: 実行

図 16 のようにメッセージペインに”Hello World”と表示されたらプロジェクトの完了です。

図 16: 終了コード 0

エクスプローラーで作業用のフォルダーをみると図 17 のように”first.exe”が作成されているのが確認できます。

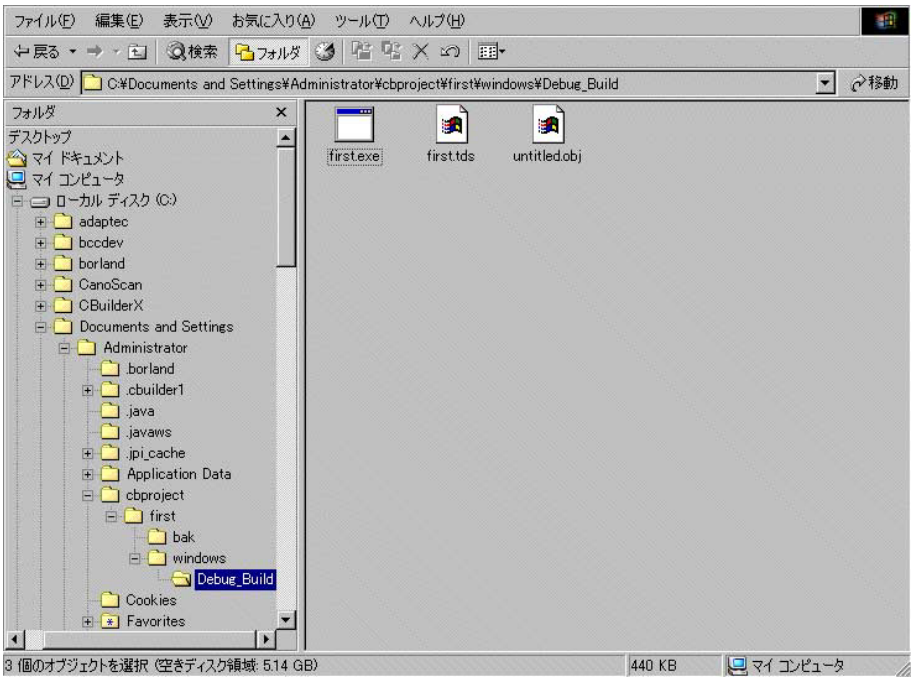


図 17: first.exe

3 C言語の基本的なルールとグラフ作成の方法

3.1 幾つかの約束事

最初に簡単な計算を試みる。以下のように内容ペインに記述する。

```
#include <stdio.h>
#include <math.h>

double x,y;

int main()
{
    x=2;
    y=sqrt(x);
    printf("%f のルートは %f です。\n",x,y);
    return (0);
}
```

#include <stdio.h>は標準的な入出力関数が定義されているパッケージです。この中にはprintf 関数などが含まれています。#include <math.h>はルートなどの数学関数が定義されているパッケージです。定義されている関数には他に下記のようなものがあります。

```
sin(x)   : x の正弦
cos(x)   : x の余弦
tan(x)   : x の正接
exp(x)   : 指数関数
log(x)   : 自然対数
log10(x) : 常用対数
sqrt(x)  : 平方根
fabs(x)  : 絶対値
```

double x,y; は変数 x と y が倍精度の実数型であることを宣言しています。このようにC言語では実数と整数を区別して取り扱い、また変数を使用する前にデータの型を宣言する必要があります。データの型には下記のようなものがあります。

```
int    : 整数型
float  : 単精度実数型
double : 倍精度実数型
```

int main() 以降がプログラムの本体です。 x=2 は代入文といわれ、変数 x に 2 を与えます。次行の y=sqrt(x) も代入文ですが、この場合は x のルートを計算した値を y に代入します。次行の printf 文は画面に計算結果を表示します。printf の働きをみるために、ここでプログラムをコンパイルして実行するとメッセージペインに下記のように表示されます。

```
2.000000 のルートは 1.414214 です。
```

printf 文中にある最初の%f は x を倍精度の実数として表示せよという指示をあらわしています。整数を表示するときには%d を用います。return(0) という行はオペレーティングシステムに終了のメッセージを伝えるものです。

3.2 配列とループ

運動方程式を数値的に解く場合には、下記のような一定時間ごとの座標値を求めることになる。

$$x(t_0), x(t_1), x(t_2), x(t_3) \cdots$$

このような一連のデータを取り扱う場合には、配列という、変数に添え字を付けたものを利用する。10 個の配列であれば、下記のように宣言する。

```
double x[10];
```

ここで注意すべきことは、添え字が 0 から始まる点である。そのため x[0] から x[9] までが利用できる。

座標値を計算するには同じ計算過程を繰り返すことが必要になるが、ここでは繰り返しの制御文として for ループを用いる。下記のプログラム中の i はカウンタと呼ばれる制御用の変数で i=0 から始まって 1 ずつ増加させて 9 になるまでループ内の命令を繰り返す。i+=1 は i=i+1 と同じ意味である。

```
for(int i=0; i<=9; i+=1){
   t[i]=2*3.1415*(i/9);
   x[i]=sin(t[i]);
}
```

ここで、配列とループを用いてプログラムを作成してみる。下記のように内容ペインに記入する。#define PI 3.14159 は PI を 3.14159 で置き換えるマクロである。

```
#include <stdio.h>
#include <math.h>

#define PI 3.14159

double x[10],t[10];

int main()
{
  for(int i=0; i<=9; i+=1){
       t[i]=2.0*PI*(i/9.0);
       x[i]=sin(t[i]);
       printf("i= %d , x = %f \n",i,x[i]);
  }
    return (0);
}
```

コンパイルして実行するとメッセージペインに下記のように表示される。

```
i= 0 , x = 0.000000
i= 1 , x = 0.642787
i= 2 , x = 0.984808
i= 3 , x = 0.866026
i= 4 , x = 0.342022
i= 5 , x = -0.342017
i= 6 , x = -0.866024
i= 7 , x = -0.984808
i= 8 , x = -0.642791
i= 9 , x = -0.000005
```

3.3 グラフを書く

計算結果をグラフで表してみる。C 言語にはグラフ描画用の命令がないので、ここでは”GLIBW32 ver1.36” というライブラリーを利用する。

(作成 Yamada. K 氏, http://www.asahi-net.or.jp/ uc3k-ymd/)

内容ペインに下記のように記入する。

```
#include <stdio.h>
#include <math.h>
#include "glibw32.h"

#define PI 3.14159

double x[10],t[10];

int main()
{
  for(int i=0; i<=9; i+=1){
       t[i]=2.0*PI*(i/9.0);
       x[i]=sin(t[i]);
       printf("i= %d , x = %f \n",i,x[i]);
  }
ginit(450,350,WHITE);

GRAPH g;

char strl[]="Sin 関数";
textout(10,5,strl,BLACK,1,1);

g.window(0,-1.2,2*PI,1.2);
g.view(20,20,420,320);
g.axis(0.2,0.1);
g.setcolor(RED);

for( int i = 0; i<=9; i+=1){
    g.fcircle(t[i],x[i],0.04);
}
savebmp("myfile.bmp"); // gend() の直前に書くこと！
gend();

return (0);
}
```

ginit(450,350,WHITE) 以降がグラフを書く部分である。以下簡単に説明をする。

`ginit(450,350,WHITE);`	サイズ 450x350 ピクセルのグラフ窓を用意
`GRAPH g;`	GRAPH ブジェクトのインスタンス化
`char strl[]="Sin 関数";` `textout(10,5,strl,BLACK,1,1);`	窓に文字を書く
`g.window(0,-1.2,2*PI,1.2);`	イメージする論理座標を決める
`g.view(20,20,420,320);`	描画するスクリーン座標の範囲を決める
`g.axis(0.2,0.1);`	座標軸の目盛りを設定
`g.setcolor(RED);`	描画の色を変更

`for( int i = 0; i<=9; i+=1){`
`  g.fcircle(t[i],x[i],0.04);`
`}`
	円を描いて塗りつぶす。0.04 は円の半径

`savebmp("myfile.bmp");`	グラフをファイルとして保存
`gend();`	描画終了

この他の機能として、画面に点を打つ g.pset(x,y) 等があります。詳細はデスクトップ上にある glibw32 の” 関数仕様 HLP” を参照してください。

コンパイルをする前に `glibw32.lib` を下記の手順でプロジェクトに追加する。プロジェクトペインで右クリックして [ファイルの追加] を選び (図 18)、図 19 で下記のファイルを選択する (作業用のフォルダーにはありません)。

ファイル名　　`C:\CBuilderX\lib\glibw32.lib`

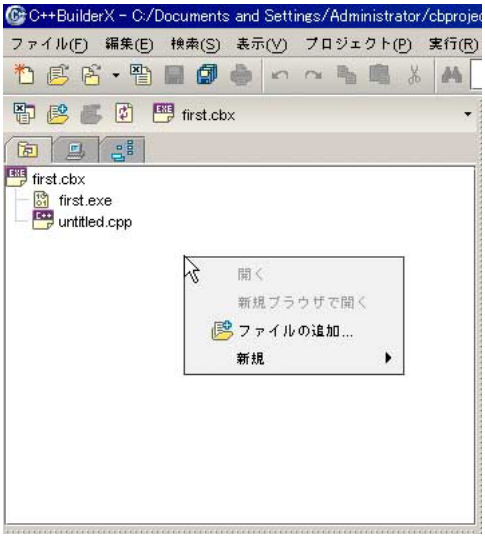


図 18: ライブラリーの追加

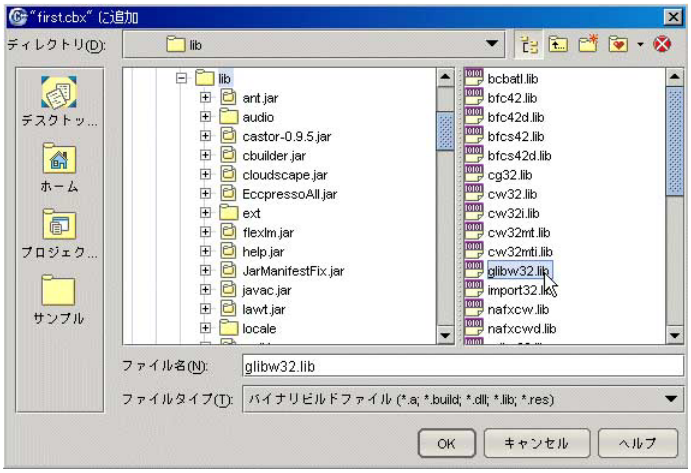


図 19: ライブラリーの選択

追加されたライブラリーファイルはプロジェクトペインに現れる (図 20)。

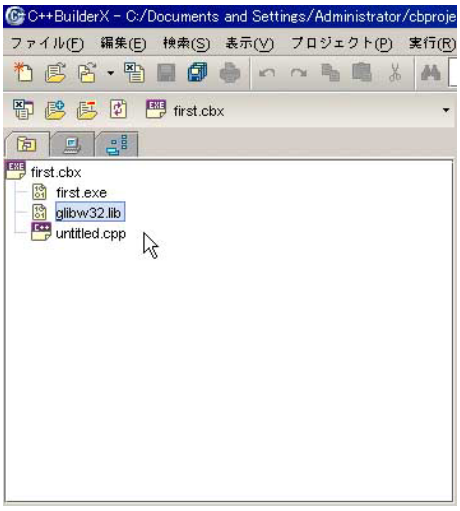


図 20: 追加されたライブラリー

これでライブラリーを利用する準備が整ったので、プログラムをコンパイルする。ただしプログラムの実行はツールバーの [プロジェクトのデバッグ] をクリックする (図 21)。

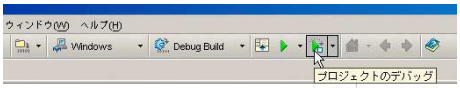


図 21: プロジェクトのデバッグ

コンソール窓 (黒い画面) とグラフ画面が表示される (図 22)。これと同時に下記のファイルが作成される。

```
C:\Documents and Settings
        \Administrator\cbproject\first\windows\Debug_Build\myfile.bmp
```

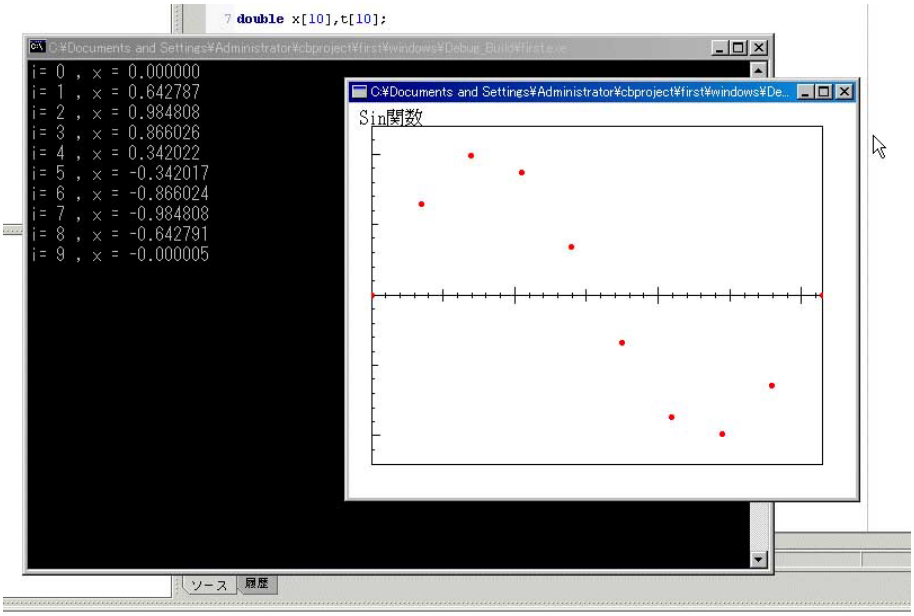


図 22: グラフの表示

4 重力場における運動

運動方程式を数値的に解く例として、重力場における上方に投げあげられた物体の運動を考えてみよう。鉛直上方に投げあげるとして、時間を t[s]、高さを x[m]、速度を v[m/s] で表し、初期条件として $t = 0$ で速度を v_0、高さを 0[m] とする。また重力加速度を g[m/s^2] とする。

この場合の運動方程式は物体の質量を m[kg] として、

$$m\frac{dv}{dt} = -mg$$

よって Δt 経過したときの速度変化 Δv は

$$\Delta v = -g\Delta t$$

よって時刻 $t + \Delta t$ での速度は

$$v(t + \Delta t) = v(t) + (-g\Delta t)$$

また Δt 間の移動距離 Δx は

$$\Delta x \simeq v\Delta t$$

以上をプログラミングで表現してみよう。

4.1 最大値を求める

下記のように内容ペインに記入する。使用した変数は t，v，x がそれぞれ時間、速度、座標である。ひとつのステップの時間間隔を 0.01sec として 4sec まで計算している。初期条件は $t = 0$ で x[0] = 0、v[0] = 9.8 である。以下先のステップの速度と座標を用いながら次のステップの速度と座標を計算している。

```
#include <stdio.h>
#include <math.h>
#include "glibw32.h"

#define SIZE 401

double t[SIZE],tmax=-1000000,tmin=1000000;
double v[SIZE];
double x[SIZE],xmax=-1000000,xmin=1000000;
double dt=0.01,g=9.80;

int main()
{

  t[0]=0.0;
  x[0]=0.0;
  v[0]=9.80;
  for(int i=1; i<=(SIZE-1); i+=1){
       t[i]=dt*i;
       v[i]=v[i-1]+(-g)*dt;
       x[i]=x[i-1]+v[i]*dt;
  }

for(int i=0; i<=SIZE-1; i+=1){
if(t[i]>tmax)(tmax=t[i]);
if(t[i]<tmin)(tmin=t[i]);
if(x[i]>xmax)(xmax=x[i]);
if(x[i]<xmin)(xmin=x[i]);
}

printf("tmax = %f tmin = %f \n",tmax,tmin);
printf("xmax = %f xmin = %f \n",xmax,xmin);

ginit(450,350,WHITE);
```

```
GRAPH g;

char strl[]="放物線";
textout(10,5,strl,BLACK,1,1);

g.window(0,1.2*xmin,1.1*tmax,1.2*xmax);
g.view(20,20,420,320);
g.axis(tmax/10,(xmax-xmin)/10);
g.setcolor(RED);

for(int i = 0; i<=SIZE-1; i+=1){
    g.pset(t[i],x[i]);
}
savebmp("myfile.bmp"); // gend() の直前に書くこと！
gend();

return (0);
}
```

グラフを作図するには時刻と座標の最大値最小値を知る必要がある。このためここでは `if` 文を利用している。

```
if(x[i]>xmax)(xmax=x[i]);
```

上記の `if` 文は引き続く（ ）の中の条件が成り立つならば (`x[i]` が `xmax` よりも大きいならば)、その後に続く（ ）の中の命令を実行する (`xmax` を `x[i]` で置き換える)。

コンパイルして実行すると下記のグラフが得られる。

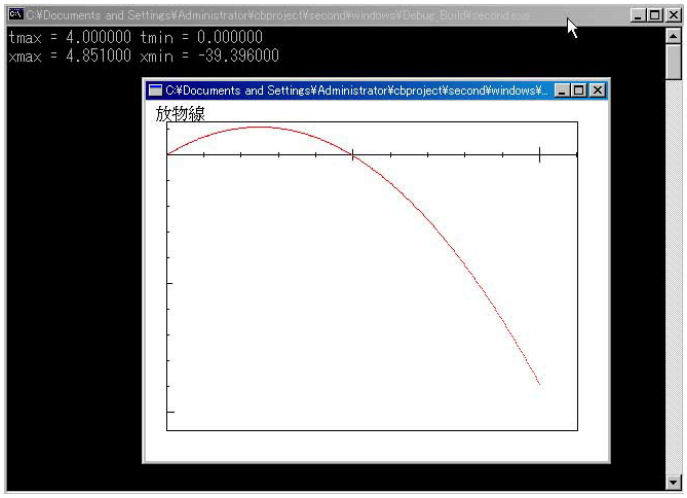


図 23: 放物線

4.2 課題１

下記の運動方程式で表される運動が減衰振動となることを示しなさい。x[m] は質点の座標、v[m/s] は速度、k は正の定数であり、速度に比例する摩擦力 $-\mu v$ が働くものとする。$m = 1.0$,$k = 1.0$,$\mu = 0.2$ と定めよ。また初期条件は $t = 0$ において $x = 1$,$v = 0$ とする。

$$m\frac{dv}{dt} = -kx - \mu v$$

また運動方程式を下記のように変更した場合はどのような運動となりますか。$\eta = 0.1$, $\omega = \sqrt{k/m}/20$ を用いること。

$$m\frac{dv}{dt} = -kx - \mu v + \eta\frac{\sin\omega t}{|\sin\omega t|}$$

5　プログラミングによる機器の制御

以下では、プログラミングによる機器の自動制御を取りあげる。カノープス株式会社製の USBit を用いてラジコンカーをコントロールしてみる (図 24 の左にあるものが USBit)。

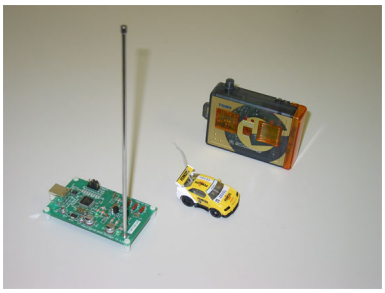


図 24: USBit

5.1　1 秒間の直進

新規にプロジェクトを作成し、USB 接続の送信装置を動作させるために、プロジェクトペインに下記のライブラリーを追加します。

C:\CBuilderX\lib\USBitLib_OMF.lib

内容ペインに下記のように記入します。このプログラムは車を 1 秒間直進させて停止させるものです。

```
#include <windows.h>
#include <stdio.h>
#include "USBitLib.h"

long int tick0,tick;

void main()
{
  HANDLE hUSBit = NULL;
  USBIT_STATUS status = USBit_Initialize(&hUSBit, USBIT_KIND_BITCHARG, 0 );
  if (status != USBIT_STS_NOERROR || hUSBit == NULL )
  printf("初期化できませんでした。 \n");

  USBit_SetCommand(hUSBit, USBIT_CMD_FORWARD, 0 );

  tick0 = GetTickCount();
  tick =tick0;
  while (tick-tick0 < 1000){tick = GetTickCount();
                            }

  USBit_SetCommand(hUSBit, USBIT_CMD_STOP, 0 );

  USBit_Destroy(hUSBit, 0 );
}
```

`#include "USBitLib.h"` は送信機を制御する関数の宣言文です。関数には下記のものがあります。

```
USBit_SetCommand(hUSBit, USBIT_CMD_BACKLEFT, 0 )  ;   左後方
USBit_SetCommand(hUSBit, USBIT_CMD_BACKWARD, 0 )  ;   後進
USBit_SetCommand(hUSBit, USBIT_CMD_BACKRIGHT, 0 ) ;   右後方
USBit_SetCommand(hUSBit, USBIT_CMD_STOP, 0 )      ;   停止
USBit_SetCommand(hUSBit, USBIT_CMD_FORLEFT, 0 )   ;   左前方
USBit_SetCommand(hUSBit, USBIT_CMD_FORWARD, 0 )   ;   前進
USBit_SetCommand(hUSBit, USBIT_CMD_FORRIGHT, 0 )  ;   右前方
```

`main` 以下最初の 4 行は送信機を初期化する命令です (このまま記入すること)。

時間の計測には GetTickCount() 関数を使用します。#include <windows.h>は、この関数が定義されている宣言文です。時間はミリ秒単位の整数値として得られます。計測した時間を代入する変数 tick,tick0 は long int(大きな整数) としてデータ型を指定します。

1 秒間という時間の間隔を作り出すために、while() ループを用いています。これは ( ) 内の条件式が成立するまで続く{ }内の命令を繰り返し実行するもので、時間の初期値 tick0 から 1000msec 経過するまで時間計測を繰り返します (tick0 と tick との差が 1000 になるまで)。

プログラムをコンパイルした後、コンピューターの USB 端子に USBit が接続されているのを確認してラジコンカーを走らせて見てください。最初はラジコンカーは充電されていませんので、備え付けのファイルを参照して充電してください。

5.2 課題２

用意してある障害を回って８の字走行をするプログラムを作成すること。

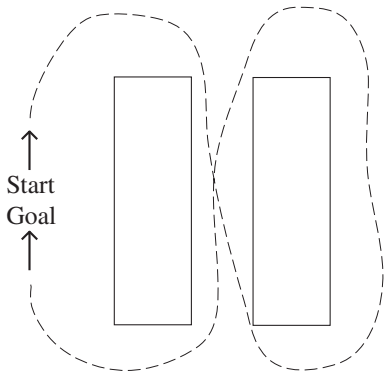


図 25: 8 の字走行

5.3 課題３

走行と同時に効果音を鳴らすこと。音を再生する場合には，まずプロジェクトペインに下記のライブラリーを追加します。

```
C:\CBuilderX\lib\psdk\winmm.lib
```

内容ペインに下記のように記入すると sample.wav ファイルが再生されますので，参考にしてください。音声ファイルはプログラムと同じフォルダーに置いてください。効果音のサンプルがデスクトップ上の USBit_Sound フォルダーに在ります。

```
#include<windows.h>
#include<mmsystem.h>
void main()
{
PlaySound("sample.wav",NULL,SND_FILENAME | SND_SYNC);
return;
}
```

6　写真資料

6.1　減衰振動

下記の図は電気回路上の減衰振動です。課題 1 の結果に相当します。

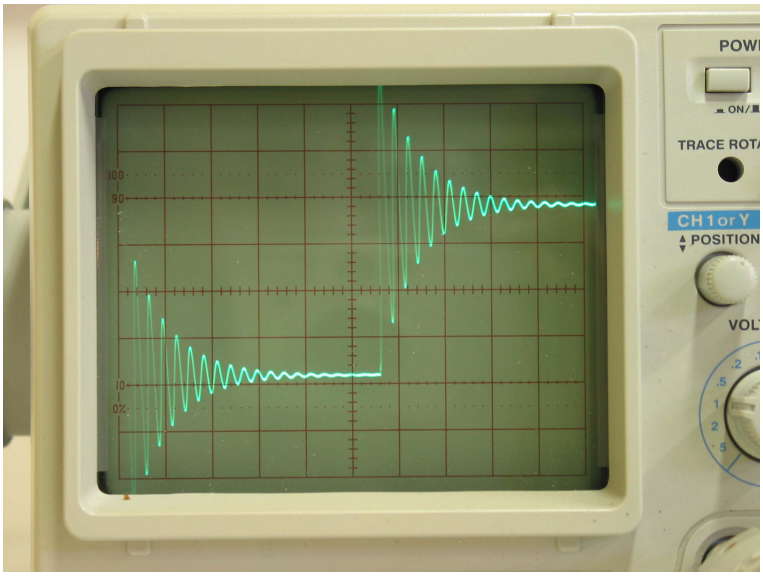


図 26: 減衰振動その 1

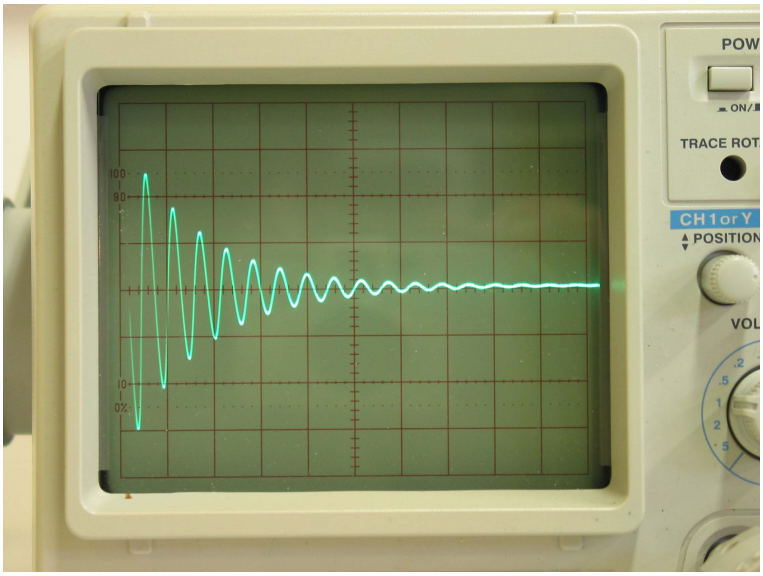


図 27: 減衰振動その 2

6.2 ラジコンの映像

下記の図は課題 2 と 3 で用いるラジコンの映像です。下図は USB 接続の発信器です。

図 28: ビットチャージ

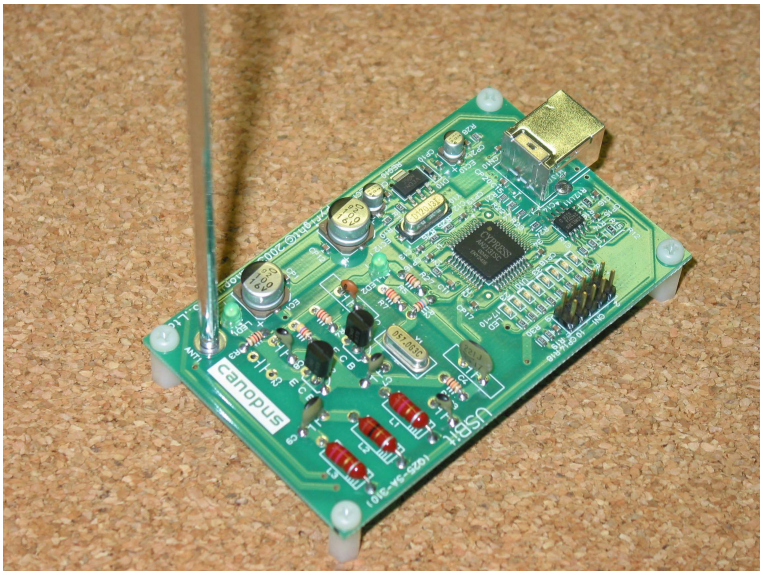


図 29: USBit